Calculating Costs for Quality of Security Service

Evdoxia Spyropoulou
Anteon Corporation
Monterey, CA

Timothy Levin
Anteon Corporation
Monterey,CA

Cynthia Irvine
Naval Postgraduate School
Monterey, CA

Abstract

This paper presents a Quality of Security Service (QoSS) costing framework and demonstration. A method for quantifying costs related to the security service and for storing and retrieving security information is illustrated. We describe a security model for tasks, which incorporates the ideas of variant security services invoked by the task, dynamic network modes, abstract security level choices and resource utilization costs. The estimated costs can be fed into a resource management system to facilitate the process of estimating efficient task schedules. Integration and scalability issues have been taken into account during the design of the QoSS costing demonstration, which we believe is suitable for incorporation into a resource management system research prototype[1].

1. Introduction

The introduction of metacomputing and distributed resource management mechanisms to the Internet and World Wide Web will make available to users and applications a large diversity of previously unavailable network and computing resources. Middleware resource management systems (RMSs) will use geographically distributed, heterogeneous resources to support applications with a wide range of computation needs [3][2][5][4]. The RMS in such an environment is responsible for: efficiently scheduling multiple simultaneous tasks onto specific network resources; supporting user requirements for performance and security; and providing support for tasks to adapt to changing resource availability.

As part of the process of estimating efficient task schedules, the RMS must balance resource-usage costs against user benefits, since there might not exist sufficient resources to maximize the benefits to all users. Thus the RMS must be able to quantify the costs associated with the entire range of network services. Various research efforts address these issues. Huh et al [5] present a method for unifying dynamic resource requirements among heterogeneous hosts of a RMS, using a system resource model with initial profiles for applications. A scheduler for a RMS is presented by Dail et al [2] along with its performance model, which requires “good predictions of megabytes transferred, number of messages initiated, overhead factor, benchmarks for program CPU and memory utilization over the different target architectures”.

Costing of security services in this context has received little attention. Inherently, Quality of Service (QoS) involves user requests for (levels of) services, which are related to performance-sensitive variables in an underlying distributed system. For security to be a real part of QoS, then, security choices must be presented to users, and the QoS mechanism must be able to modulate related variables to provide predictable security service levels to those users.

The notion of security variability has been discussed before. A Quality of Protection parameter is provided in the GSS-API specification [11]. This parameter is intended to manage the level of protection provided to a message communication stream by an underlying security mechanism (or service). Another early reference to a variable security service is that of Schneck and Schwan [12], which discusses variable packet authentication rates with respect to the management of system performance.

The challenge is to associate costs with the entire range of network security services, and provide these costs to a RMS scheduler. In previous work we have discussed fundamental Quality of Security Service (QoSS) concepts in terms of variant security mechanisms and dynamic security policies [6], along with an analysis of the layered and variable security services and requirements presented to a RMS [8]. In [9] we defined a preliminary security service taxonomy defining the range of security services a RMS may need to manage. [10] addressed the problem of how users and administrators can understand and easily interact with the wide range of security services and mechanisms, by providing methods

[1] This work was sponsored by the DARPA/ITO Quorum Program under contract number .The views expressed in this paper are those of the authors and do not reflect the official policy of the Department of Defense of the U.S. Government

Report Documentation Page

Form Approved
OMB No. 0704-0188

Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

1. REPORT DATE **DEC 2000**	2. REPORT TYPE **N/A**	3. DATES COVERED **-**

4. TITLE AND SUBTITLE **Calculating Costs for Quality of Security Service**	5a. CONTRACT NUMBER
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Naval Postgraduate School Department of Computer Science Monterey, CA 93943**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**UU**	**10**	

for translation of a simplified user abstraction of security to detailed underlying mechanisms.

Using the base provided by our previous research, we present in this paper a QoSS costing framework and demonstration, which illustrates how costs associated with network security services can be calculated and supplied to a RMS. The method for modeling variant security characteristics of applications and for computing relevant costs and the method's implementation, are presented here. Through the use of high-level interfaces –an administrative one for representing dynamic network policies, and a user interface for selecting security levels for the application– we demonstrate the way that security could be treated as a QoS dimension.

The remainder of this paper is organized as follows: Section 2 describes security offered to tasks in the network context. In Section 3 the notion of variant security and of security as a dimension of QoS is presented. Section 4 discusses briefly the use of a security vector for the description of network security policies. In Section 5 we explain the idea of dynamic network policies associated with the system's mode. The interaction with users for selection of security level for tasks is discussed in Section 6. Section 7 explains our approach for calculating costs related to security. In Section 8 we present in detail the QoSS Costing Demonstration; and a conclusion follows in Section 9.

2. Tasks and Security Services in a Network System

A network system is defined as the totality of network-accessible resources. In the network computing context, users or user programs may request the execution of applications/tasks, which are scheduled by an underlying control program to execute on local or remote computing resources. A task's utilization of various network services and resources may be intermediated by different QoS middleware mechanisms (QoSM). Thus, a task is invoked in a sequence:

- The user activates the application through some interface with an application manager.
- The application is intermediated by the QoSM.
- The QoSM submits the application to the system [6].

The execution of the task may access or consume a variety of resources such as: local I/O device bandwidth, internetwork bandwidth; local and remote CPU time; local, intermediate (e.g., routing buffers) and remote storage.

The application on the network is presented with various security services: this means that (at some point or) during its execution it may utilize certain security services. A *security service* is a high-level abstract resource providing security functionality such as: authentication, auditing, privacy, integrity, intrusion detection, non-repudiation and traffic flow confidentiality [7]. A security service typically consumes other low level system resources, and may be implemented by one or more security mechanisms.

In [9] a preliminary security service taxonomy is presented with example mechanisms for each service. Each security mechanism is associated with a *service area*, which indicates the general topographical component of the network in which the security or protection is effective. The taxonomy identifies three service areas: end system (e.g., a client or server system), intermediate node (e.g., routers, switches), and network connection (i.e., the "wire" connecting various systems and nodes). An additional total subnet service area identifies mechanisms that cannot be assigned exclusively to either of the above areas (e.g., boundary control mechanisms).

3. Quality of Security Service and Variant Security

For a Quality of Service (QoS) dimension to be supported means that users can request or specify a level of service for one or more attributes of this dimension, and the underlying QoS control mechanism is capable of entering into an agreement to deliver those services at the requested levels [1][15]. Therefore, the control mechanism must be able to modulate the level of the service to individual subscribers (e.g., users).

Users may have expectations (i.e., functional and assurance requirements) with respect to the security services they are provided. Quality of Security Service (QoSS) has the meaning that security and security requests can be managed as a responsive "service" for which quantitative measurement of service "efficiency" is possible [8].

QoS mechanisms can be more effective with security appearing as a QoS dimension: when variable levels of security services and requirements are presented to users or network tasks, the underlying system can adapt more gracefully to changes in resource availability during the execution of a task, and thereby do a better job at maintaining requested or required levels of service in all of its dimensions.

The enabling technology for both QoSS and a security-adaptable infrastructure is variant security, or the ability of security mechanisms and services to allow the amount, kind or degree of security to vary, within predefined ranges. This notion of network Quality of Security Service has the potential to provide administrators and users with more flexibility and potentially better service, without compromise of network and system security policies.

To be general, we will define that all security requirements have a range of permissible behavior. In some cases, a range may be unitary, or degenerate, in which case it represents no choice. Where a range represents a choice, the requirement is termed *security variant* [6]. In the same sense all system security services can be considered as having a range: since they are invoked at the discretion of the user or application, the range is at least binary (i.e., invoked or not invoked).

This notion of variant requirements and security ranges may, at first, seem strange. For many, either you have security or you don't. This is true on a gross scale, since without some minimum level of security, a system will be considered inadequate for user requirements. But if a user's minimum requirements are met, there can be some choice with respect to what is adequate.

Some examples in which security ranges and choices could be available:

- collaborative applications, such as video teleconferencing with shared electronic boards and application suites: if a group member is participating in the collaboration from a hotel room in a foreign country known for government support of corporate espionage, his security requirements and choices will be quite different than if he were in "friendly" territory. These security choices may form a range from which the user or application can select, and can include different levels of authentication, confidentiality and integrity.
- a variable packet authentication scheme [12]: the recipient might be satisfied if a certain percentage of each packet in an image stream was authenticated (e.g., 80% to 100%); this might have applicability for image display, especially considering that the low order bits of each byte are not very significant visually, in some display protocols.
- an Intrusion Detection System (IDS): an administrator may choose to run the IDS within a range rather than a fixed level. There would be a minimal level of IDS processing below which the system would not be permitted to fall, but the IDS would be balanced against performance requirements of the organization's tasks. Thus the IDS might perform more thoroughly (with deeper histories) when the system is lightly loaded than during peak hours. The administrator might also choose to set an upper limit to IDS performance.

The following are some example security variables [7], with characterizations of how they could be specified or measured:

- Strength of cryptographic algorithm, e.g., RSA, DES: measured in terms of the work factor associated with a brute force attack
- Length of cryptographic key: characterized by bit-length
- Percentage of packets authenticated: characterized by percentage of total (e.g., a multimedia environment might tolerate a percentage of data modification or loss)
- Security functions present in destination job-execution environment: characterized by operating system or boundary control security policy enforcement mechanisms.
- Confidence of policy-enforcement in remote login environment: characterized by third party evaluation
- Robustness of authentication mechanism: here the range might span weak password, strong password, biometric, and smart cards with on-board display and input interfaces.

4. Security Policy and Security Vector

The body of rules for resolving network security issues is called the *network security policy*. These policies consist of broad policy jurisdictions, which can be decomposed typically, into functional requirements. Previous work [8] represents the n-dimensional space of functional security requirements involving a task executing in a network environment, as a vector (S) of security components. Each component (S.c) specifies a boolean functional requirement, and it may indicate positive requirements as well as negative constraints. By convention a component S.c may contain only one variant clause, i.e. one security variable.

Examples of security vector components are:

S.1: level(user) >= level(resource)

S.2: length of confidentiality encryption key >= 64,
<= 256

S.3: % packets authenticated >= 50, <=90

Requirements for a specific security service may be represented by one or more components, indicating a *service sub-vector*. A task executed in the network environment may invoke many security services. The union of service sub-vectors, for all the task's services, forms the task's security vector S.

When the number of possible choices indicated in a component S.c is greater than one, we talk about a variant security component, since the underlying control program has a range within which it may allow the job to execute with respect to the policy requirement.

A range is a set of elements, which define the possible choices for a security variable [6]. The elements of a range are related, because some are more secure than others, providing thus a means for ordering them. With respect to relative security, encryption algorithms may be ordered by crypto-analytic work factor, subsets of a network may be partially ordered by set inclusion, a packet encryption percentage can be linearly ordered based on numeric value.

5. Network Mode and Dynamic Security Policies

A task is characterized by a set of security requirements that must be met. The network operational status or "mode" could influence the security restrictions and available security services for the task, because under certain conditions, the user or administrator may be willing to accept more (or less) security for a given application. For example, during an emergency, a military commander might decide to forgo certain security protocols in order to get some important information transmitted quickly. This decision changes the security policy, but the actual policy arrived at may not be clearly understood.

With a dynamic security policy, the security restrictions and available security policies allow their functional requirements and implementation mechanisms to be examined with respect to the overall policy, prior to being fielded, rather than depending on an ad hoc review [10]. If dynamic policies are created before deployment of the computer network, the network can respond to changing environments, by having access to a predefined set of alternate security policies. For example, a corporate intranet might have a mode indicating that the system is under attack from the internet. In this mode, it might be desired for a higher degree of network security to be in place. Or, an Internet Service Provider might receive a large amount of simultaneous requests and enter an "impacted" mode, in which certain optional security services would be curtailed for efficiency. In each of these cases, the effects of changes to the security mechanisms would be predefined and limited to meet the desired alternate security policy.

With a dynamic security policy, the acceptable range for the security variable of a security vector's component depends on the network "mode"[8]. We refer to three example modes: normal, impacted, emergency. Representing with S^{mode} a separate security vector for each operational mode, components could be assigned different values, for example:

$S^{normal}.a$: length of confidentiality encryption key >= 64, <= 256
$S^{impacted}.a$: length of confidentiality encryption key >= 64, <= 128

Or, for example, policy makers might decide that the policy should remain in force regardless of the network mode:

$S^{normal}.b = S^{impacted}.b = S^{emergency}.b$:
clearance(user) = classification(resource)

The network policy can be expressed through the allowable range of values for the requirement components. Assuming that a security variable A can take all values from 0 to 1 (e.g., a packet authentication rate), Figure 1 shows the policy's interpretation for the values of the variable in each mode. In normal mode we accept a range of values from 0.5 to 1, in impacted mode we move to lower values, whilst emergency means high security without many choices.

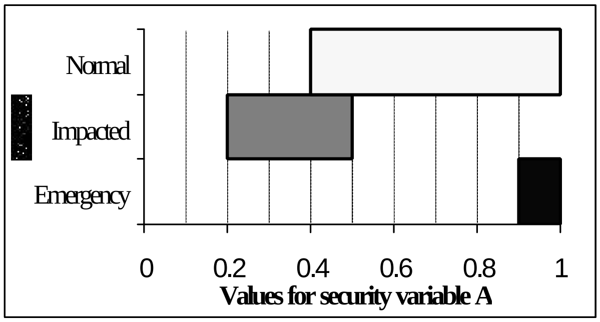


Figure 1: A security policy interpretation for different network modes

A different approach to what each mode would mean for the system, is expressed by Figure 2, where emergency mode is closer to a "battle short" approach, in which to accomplish the mission as quickly as possible the security variable's value must be low. In normal mode we offer a generally higher level for this security variable.

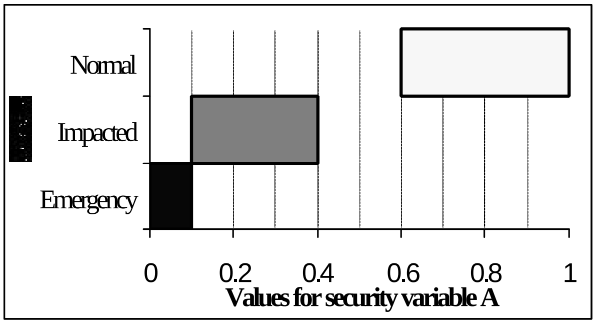


Figure 2: An alternative interpretation for network modes

Ranges for different modes can have overlapping values, or certain values may be completely excluded from the permissible set of values of any mode for a security variable. Furthermore the range need not be continuous (e.g. there could be a step of increment, sets of discrete values).

6. User Choice for Security Level

The over-arching network security policy demands some minimum levels of security service for a task, indicating also the maximum security levels that can be provided by the system. Selections for QoSS may be provided to users to any degree of security within these limits. Thus, a system can always provide more security, at the user's discretion, than the minimum required by the base security policy, while still complying with the policy.

Still, the security services and underlying mechanisms may present too many variables and choices for users or applications to manage without automated support. Instead of presenting to the user all combinations of security mechanisms and parameters for the variant services, we can offer a simplified abstraction of security, in the form of security level choices, like "high", "medium", "low" (e.g. as is commonly seen now in web browsers and personal firewalls).

The elements of the simple user interface can be mapped to detailed mechanism invocations via a translation matrix [10]. The security administrator or system security engineer would pre-select various specific mechanisms and settings that are assigned to the security variables for each of the three choices offered to the user.

Figure 3 shows this translation of abstract security level to predefined settings for the variable A of the previous paragraph, assuming we are on normal mode.

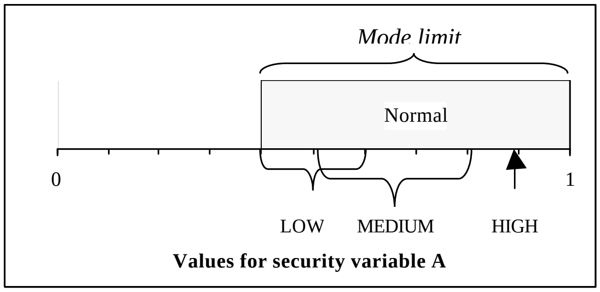


Figure 3: Security variable's A settings in normal mode for abstract security level choices

A security level can be mapped to a sub-range within the acceptable range, or could be mapped to a specific value. In the former case the underlying RMS would be responsible for assigning security services and resources to the user that would meet the security profile indicated by the translation matrix. If corresponding services or resources could not be found to meet user request, then the RMS would need to negotiate different degrees of service with the user, or perhaps use a default translation.

7. Costs for Variant Security

Why would a user request anything less than the highest level of security? The answer is cost. As with multimedia image resolution, users will generally desire the greatest amount of security (or image fidelity) available, but this desire is tempered by cost. Cost may take the form of monetary charges (unlimited bandwidth but at a high cost per byte) or performance degradation (for high resolution, processing and downloads times will be long), for example. When cost is very high (e.g., slow response time, image display), users may be willing to accept security (or imagery) that is less than their ideal level of service [6].

If a particular security mechanism is "fixed" (i.e., always applied) then the overhead for the mechanism is part of the normal cost of running the task and the normal costing mechanism used by the RMS will suffice. For variant security mechanisms, however, the security overhead will vary, depending on the user's QoS request. Some task invocations will utilize little, if any, of the variant mechanism and other invocations may utilize the mechanism at an increased level. Also, the scheduler may adapt security support, while maintaining any minimum system security policy requirements, in order to schedule the tasks most efficiently. The RMS must calculate how much the use of the security mechanism will increase the cost of the task, according to the specific security "level" requested. For this reason the RMS must have access to detailed information about the resource cost (as well as the task's requested QoS) for each variant security mechanism. Near-optimal solution selection for task schedules depends on the accurate estimation of per-task, per-resource, cost of security [9].

In our approach for quantifying the costs related to a task's security requests, we refer to costs relative to every security service invoked by the task. Each service may access some or all of the following resources: CPU time, memory, bandwidth (other cost factors are possible, e.g. disk space, and will be added to our framework). The resource usages may be temporary or persistent. That's why we discriminate between start-up and streaming costs.

For example the Confidentiality on the Network Connection service, using a symmetric algorithm like Twofish [13] for data encryption, would require some extra processing during start-up for the initialization of S-boxes. This is a one-time cost during the establishment phase of the service. On the other hand bandwidth costs for the confidentiality service are streaming costs only, in the form of extra bytes per packet due to the encryption algorithm.

In a QoS system every application would have its resource costs modeled as shown in Table 1, where costs expressions are functions of security variables. A security variable may participate in more than one cost expression.

We refer to CPU costs in clocks (or clocks/packet), memory costs in bytes, bandwidth costs in bytes (or bytes/packet). In another approach, all measures could be unitless and normalized within a common framework. A careful description of the semantics of the units with respect to each security service would then be required.

Table 1: Model for security resource costs of an application

Task's Security Costs Task's Security Services	CPU costs		Memory costs		Bandwidth costs	
	Start-up	*Streaming*	*Start-up*	*Streaming*	*Start-up*	*Streaming*
Service 1	cost expression	…	…	…	…	cost expression
…	…	…	…	…	…	…
Service N	cost expression	…	…	…	…	cost expression

The cost expressions and units we use for our research are preliminary. Our purpose is to demonstrate how variant security can be modeled and the related costs quantified. The intended use of the cost results is to supply needed information to a RMS for efficient scheduling of tasks, treating security as a QoS dimension.

In the next Section, the QoSS Costing Demonstration is an implementation paradigm for security costs' quantification. The cost expressions used should be considered as placeholders. Continued effort is required to determine the best formulas for calculating resource costs for security services.

8. SecurityCosts: a QoSS Costing Demonstration

The Quality of Security Service Costing Demonstration illustrates the concepts described so far and provides a method for quantifying costs related to the security service. Using the taxonomy we identify services (discriminating between service areas) that tasks may invoke, and security mechanisms that implement them. We pre-define sets of security settings, corresponding to network modes and user security level choices. Costs for a task are calculated and expressed in terms of resources. These costs depend on the selected security characteristics of the specific task, and they can be fed into a RMS to facilitate the process of estimating efficient task schedules. The ongoing accuracy of cost formulas could be ensured by comparison with continuously updated historical data of resource usage kept by the RMS [4]

The implementation approach and the performance of the QoSS Costing Demonstration make it suitable for incorporation into a research prototype of a RMS. The RMS could request presentation of costs in any layout suitable for its processing. For example:

- costs for all security services, resources and cost types
- costs for a specific resource for all security services
- initialization or streaming costs only, for one (or all) resources
- costs for a particular security service of the task.

The current implementation works on a "one-time request" basis. This means that one specific costing service is requested each time, the request is processed, and results are presented. For subsequent requests the cycle repeats exactly the same. Scalability issues have been taken into account, so future work will address multiple non-sequential requests for estimation of security costs for tasks.

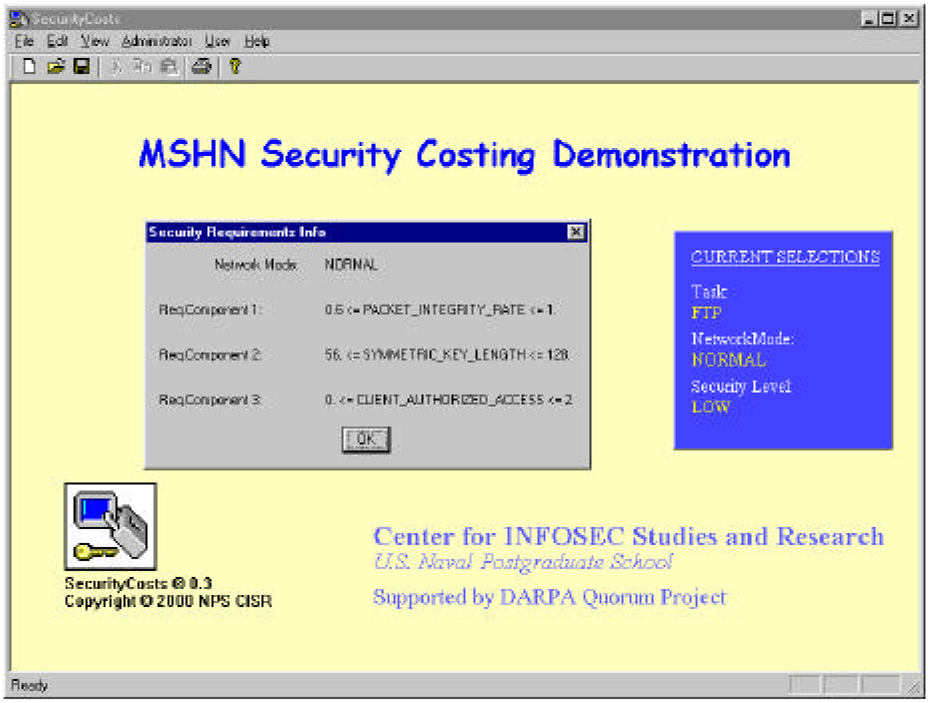
8.1 Concept of Operation

A common sequence of actions in the QoSS Costing Demonstration would be:

- The user selects from a given list the application/task, which he wishes to submit for execution. Then all internally modeled security information relevant to the application is loaded to system memory.
- The network's current mode is initially set to a default value (normal). The network administrator, depending on present system status can change this value. The specific security vector of requirements for this mode is activated, defining thus the range within which security variables can vary.
- Then the user indicates the desired security level he wishes to be applied during the processing of the task. With this selection, specific values (within the ranges accepted by the policy) are assigned to each security variable.
- The user is then ready to request processing of costs. This action actually plugs the values of security variables into the set of cost formulas that apply to the specific task's resource needs.
- Results are expressed in terms of per-service, per-resource, start-up and streaming costs.

It should be noted that the demonstration exposes some of the steps that would normally be hidden in a production system.

There is a special interface in QoSS Costing Demonstration through which the administrator can insert the model of a new task/application. What he actually



MSHN Security Costing Demonstration

Security Requirements Info

Network Mode: NORMAL

ReqComponent 1: 0.5 <= PACKET_INTEGRITY_RATE <= 1.

ReqComponent 2: 56. <= SYMMETRIC_KEY_LENGTH <= 128.

ReqComponent 3: 0. <= CLIENT_AUTHORIZED_ACCESS <= 2

CURRENT SELECTIONS
Task:
FTP
NetworkMode:
NORMAL
Security Level
LOW

SecurityCosts © 0.3
Copyright © 2000 NPS CISR

Center for INFOSEC Studies and Research
U.S. Naval Postgraduate School
Supported by DARPA Quorum Project

Figure 5: FTP security requirements for Normal mode

does is define the security services invoked by the new task, the sets of security requirements for every mode, the sets of specific values for security parameters for every mode and choice, and the cost formulas related to the task's requirements in resources.

8.2 Security Information Modeling Aspects

Various structures are used in the internal demonstration logic [14] to model the variant security characteristics of the task under cost estimation:

The set of security requirements for a task forms a *Task Requirements Vector* (TRV – see S in Section 4) and there are conceptually three TRVs, one for each mode, incorporated in the notion of a *Mode Service Matrix* (MSM).

To describe a set of specific security settings for a task, we use a *Task Variable Vector* (TVV). Availability of abstract user security level choices, leads to the need for a triplet of TVVs (one for each of the low, medium or high security choices) described by a *Choice Variable Matrix* (CVM). Furthermore the effect of network mode on the security settings is expressed through a *Mode Choice Matrix* (MCM), which contains a set of three CVMs one for each mode.

The cost for a resource during execution of a specific task is specified in a r*esource cost expression*. Costs for all resources of a service are described in a *Service Cost Vector* and costs for all services invoked by a task are associated with a certain *Cost Matrix*.

Pre-defined costing information for numerous tasks, and the security services they invoke, will be necessary if QoSS Costing Demonstration is going to work in conjunction with a RMS. This information can be stored in a database or in a set of files.

The main storage structures needed are:

- *TASK:* contains for each task indexes to corresponding Mode Service, Mode Choice and Cost Matrices.
- *MSMtoTRV:* contains for each Mode Service Matrix indexes to Task Requirement Vectors according to network mode.
- *TRV:* contains the requirement components of a task
- *MCMtoCVM:* contains for each Mode Choice Matrix indexes to Choice Variable Matrixes according to network mode.
- *CVMtoTVV:* contains for each Choice Variable Matrix indexes to Task Variable Vectors according to security level choice.
- *TVV:* contains the specific settings for the security variables of the task.
- *CM:* contains cost expressions for each resource for the security services invoked by the task.

8.3 Example

In this section we walk through the QoSS Costing Demonstration with an illustrated example.

In the beginning the user selects from a given list the application he wishes to execute in the network. Suppose he selects FTP.

It should be noted that it is not the purpose of this paper to provide an exhaustive or accurate model of FTP or other applications. The security information relevant to FTP is a combination of empirical and hypothetical data, and is used to demonstrate our ideas.

The security services (from the taxonomy) that FTP invokes are:

- Integrity on the Network Connections, since we want to assure that data travelling along the network will not be corrupted, and
- Authenticity on the End System, which means that the user may need to provide some sort of identification when connecting to the FTP site.

The security vector of requirements S for this task includes three components: the first two are associated with the Integrity on the Network Connections service, and the third is associated with the Authenticity at the End System service.

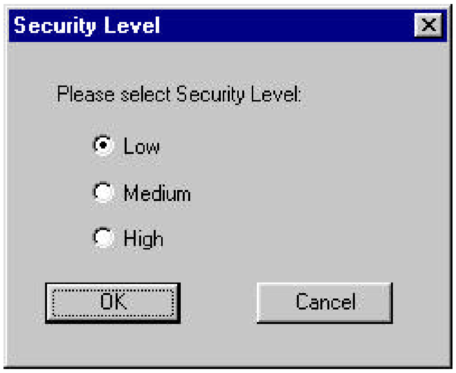
- $S_{FTP}.1$ expresses a requirement that the network communications will be cryptographically signed to provide integrity at the packet level, with a rate of integrity checking within a specific range [12].
- $S_{FTP}.2$ states that the symmetric key length will be within a certain range, assuming the use of an algorithm that accepts different key sizes.
- $S_{FTP}.3$ describes the acceptable types for access authentication. We assume there are three possible types: a user can log on without a password, he can use a simple password, or use a one-time password (crypto challenge-response). We symbolically enumerate these types as 0, 1, and 2.

The ranges for the security variables at normal mode can be seen in Figure 5.

If the mode was "impacted" for example, then the acceptable range for the $S_{FTP}.1$ requirement would be between 0.2 and 0.5, and the $S_{FTP}.2$ requirement would force the key length to be 56.

In the rest of this Section, we assume that the network is set to normal mode.

Then the user selects, through the high level interface illustrated in Figure 6, the level of security he desires to be applied during execution of the task he submits.



Security Level

Please select Security Level:

● Low

○ Medium

○ High

Figure 6: Interface for security level selection

What this selection actually does is map the high level security specification to a specific set of values to be assigned to the security variables. Table 2 shows what these values would be for all security levels.

How do our selections up to now influence the resources needs of the FTP application? Each security service invoked will probably increase the application demands for (at least some of) CPU, memory, bandwidth, either during start-up or during execution (streaming cost).

The type of cost formulas we used for the FTP task for CPU, memory and bandwidth can be seen in Tables 3, 4, and 5 respectively. In these formulas

– *KEY_LENGTH* stands for the security variable indicating the symmetric key length

– *INTEGRITY_RATE* stands for the security variable indicating the packet integrity rate

– *AUTHENTICATION_TYPE* stands for the security variable indicating the type of authorized access

– a_1, a_2, a_3, a_4, a_5, a_6, a_7, b_1, b_3, b_4, b_5, b_6, c_1 are constants

– *f(…)* indicates a non-linear dependency.

For the Integrity on Network Connections service for example, CPU start-up cost involves processing related to the symmetric key, while streaming cost is influenced by processing related to the packet integrity rate. Key length influences memory costs. A certain bandwidth–depending on the packet integrity rate– is consumed for every packet due to the Integrity service.

Table 2: Security Variables Values for all security levels in normal mode

Security Level Choice **Security Variable**	Low	Medium	High
Packet Integrity Rate	0.6	0.8	1
Symmetric Key length	56	96	128
Type of Authorized Access	No password (0)	Simple password (1)	Simple password (1)

Table 3: CPU cost formulas for FTP

FTP Security Cost Expressions	CPU costs	
FTP Sec. Services	*Start-up (clocks)*	*Streaming(clocks/packet)*
Integrity on Network Connections	$a_1 \ast$ KEY_LENGTH + b_1	$a_2 \ast$ INTEGRITY_RATE
Authenticity on End System	$a_3 \ast$ f (AUTHENTICATION_TYPE) + b_3	0

Table 4: Memory cost formulas for FTP

FTP Security Cost Expressions	Memory costs	
FTP Sec. Services	*Start-up (bytes)*	*Streaming(bytes)*
Integrity on Network Connections	$a_4 \ast$ KEY_LENGTH + b_4	$a_5 \ast$ KEY_LENGTH + b_5
Authenticity on End System	$a_6 \ast$ f (AUTHENTICATION_TYPE) + b_6	0

Table 5: Bandwidth cost formulas for FTP

FTP Security Cost Expressions	Bandwidth costs	
FTP Sec. Services	*Start-up (bytes)*	*Streaming(bytes/packet)*
Integrity on Network Connections	0	$a_7 \ast$ INTEGRITY_RATE
Authenticity on End System	c_1, if AUTHENTICATION_TYPE $\neq 0$	0

When the user requests processing of cost data the set of security variable values of the FTP task for the current mode and choice are plugged into the cost formulas. The results for normal mode can be seen in Table 6 and Table 7 for "low" and "high" security level respectively.

These results show what is generally known: there is a price to pay for high security. For example, costs for the authenticity service are 0 in the low security level, since the user logs on without a password, while at the high level, where a more sophisticated method for authentication is required, these costs are increased. Another remark is that since authorized access is granted once, there are no streaming costs for the Authenticity at the End System service. Similar observations can be made for the costs of the Integrity on Network Connections service.

9. Conclusion

In this paper a Quality of Security Service Costing Demonstration has been presented, which provides a framework for quantifying costs related to the security service and for storing and retrieving security information. A network security model for tasks has been described. In this model the ideas of variant security services invoked by the task, dynamic network modes, security level choices and resource utilization costs are incorporated.

The estimated costs can be fed into a RMS to facilitate the process off estimating efficient task schedules. Integration and scalability issues have been taken in mind during the design of the QoSS Costing Demonstration, which we believe is suitable for incorporation into a RMS prototype.

The following related areas need further research:

- determination of formulas for calculating resource costs for a range of security services
- determination of best units for cost measures
- population of the demonstration with realistic costing data
- enumeration of specific security mechanisms with respect to the described taxonomy
- organization of the security vector into a "normal" form with sub-vectors or hierarchies corresponding to security policies jurisdictions
- multiple non-sequential requests for estimation of security costs for tasks.

Our ongoing work will address these issues.

Table 6: Resource costs for FTP for LOW security level at normal mode

LOW level Security Costs **FTP Security Services**	CPU costs		Memory costs		Bandwidth costs	
	Start-up (clocks)	*Streaming* (clocks/packet)	*Start-up* (bytes)	*Streaming* (bytes)	*Start-up* (bytes)	*Streaming* *(bytes/packet)*
Integrity on Network Connections	5560	24	6200	5176	0	4.8
Authenticity on End System	0	0	0	0	0	0

Table 7: Resource costs for FTP for HIGH security level at normal mode

HIGH level Security Costs **FTP Security Services**	CPU costs		Memory costs		Bandwidth costs	
	Start-up (clocks)	*Streaming* (clocks/packet)	*Start-up* (bytes)	*Streaming* (bytes)	*Start-up* (bytes)	*Streaming* *(bytes/packet)*
Integrity on Network Connections	6280	40	6272	5248	0	8
Authenticity on End System	1200	0	69632	0	100	0

10. References

[1] Chaterjee, S., Sabata, B., Sydir, J., “ERDoD QoS Architecture”, SRI Technical Report, ITAD-1667-TR-98-075, Menlo Park, CA, May 1998

[2] Dail, H., Obertelli, G., Berman F., Wolski, R., Grimshaw, A., “Application-Aware Scheduling of a Magnetohydrodynamics Application in the Legion Metasystem”, Proc. of the Ninth Heterogeneous Computing Workshop (HCW 2000), Cancun, Mexico, May 2000, pp. 216-228

[3] Foster, I. and Kesselman, C., “Globus: A Metacomputing Infrastructure Toolkit”, Intl J. Supercomputer applications, 11(2):115-128, 1997

[4] Hensgen, D., Kidd, T., St. John. D. Schnaidt, M., Siegel, H.J., Braun, T., Maheswaran, M., Ali, S., Kim, J., Irvine, C., Levin, T., Freund, R., Kussow, M., Godfrey, M., Duman, A., Carff, P., Kidd, S., Prasanna, V., Bhat, P., Alhusaini, A., “An Overview of MSHN: The Management System for Heterogeneous Networks”, Proc. of the Eighth Heterogeneous Computing Workshop (HCW’99), San Juan, Puerto Rico, April 1999, pp. 184-198

[5] Huh, E.N., Welch, L.R., Shirazi, B.A., Cavanaugh, C.D., “Heterogeneous Resource Management for Dynamic Real-Time Systems”, Proc. of the Ninth Heterogeneous Computing Workshop (HCW 2000), Cancun, Mexico, May 2000, pp. 287-296

[6] Irvine, C. and Levin, T., “The Effects of Security Choices and Limits in a Metacomputing Environment”, Technical Report NPS-CS-00-004, Naval Postgraduate School, Monterey, CA, January 2000

[7] Irvine, C. and Levin, T., “Quality of Security Service”, to appear in Proc. of New Security Paradigms Workshop 2000, Cork, Ireland, September 2000

[8] Irvine, C. and Levin, T., “Toward Quality of Security Service in a Resource Management System Benefit Function”, Proc. of the Ninth Heterogeneous Computing Workshop (HCW 2000), Cancun, Mexico, May 2000, pp. 133-139

[9] Irvine, C. and Levin, T., “Toward a Taxonomy and Costing Method for Security Services”, Proc. of the Computer Security Applications Conference, Phoenix, AZ, December 1999, pp. 183-188.

[10] Irvine, C. and Levin, T., “A Note on Mapping User-Oriented Security Policies to Complex Mechanisms and Services”, Technical Report NPS-CS-99-08, Naval Postgraduate School, Monterey, CA, June 1999.

[11] Linn, J., Generic Security Service Application Program Interface, IETF Request for Comments: 1508, September 1993

[12] Schneck, P.A. and Schwan, K, “Dynamic Authentication for High-Performance Networked Applications”, Technical Report GIT-CC-98-08, Georgia Institute of Technology, College of Computing, Atlanta, GA, 1998

[13] Schneier, B., Kelsey, J., Whiting, D., Wagner, D., Hall, C., Ferguson N., “Twofish: A 128-Bit Block Cipher”, Counterpane Systems, http://www.counterpane.com/twofish-paper.html,June 1998

[14] Spyropoulou, E., Levin, T., Irvine, C., “Quality of Security Service Costing Demonstration for the MSHN Project”, Technical Report NPS-CS-00-007, Naval Postgraduate School, Monterey, CA, April 2000

[15] Vendatasubramanian, N. and Nahrstedt, K., “An Integrated Metric for Video QoS”, ACM International Multimedia Conference, Seattle, Wa., November 1997